Integrating Materials and Manufacturing Innovation (2024) 13:105–119
https://doi.org/10.1007/s40192-024-00341-x

THEMATIC SECTION: HARNESSING THE POWER OF MATERIALS DATA

A Common Data Dictionary and Common Data Model for Additive Manufacturing

Alexander Kuan[1] · Kareem S. Aggour[2] · Shengyen Li[1] · Yan Lu[1] · Luke Mohr[3] · Alex Kitt[3] · Hunter Macdonald[4]

Received: 6 November 2023 / Accepted: 22 December 2023 / Published online: 7 February 2024

Abstract
Additive manufacturing (AM) leverages emerging technologies and well-adopted processes to produce near-net-shape products. The advancement of AM technology requires data management tools to collect, store, and share information through the product development lifecycle and across the material and machine value chain. To address the need for sharing data among AM developers and practitioners, an AM common data dictionary (AM-CDD) was first developed based on community consensus to provide a common lexicon for AM, and later standardized by ASTM International. Following the AM-CDD work, the development of a common data model (AM-CDM) defining the structure and relationships of the key concepts, and terms in the AM-CDD is being developed. These efforts have greatly facilitated system integrations and AM data exchanges among various organizations. This work outlines the effort to create the AM-CDD and AM-CDM, with a focus on the design of the AM-CDM. Two use cases are provided to demonstrate the adoption of these efforts and the interoperability enabled by the AM-CDM for different engineering applications managed by different types of database technology. In these case studies, the AM-CDM is implemented in two distinct formats to curate AM data from NIST—the first in XML from their additive manufacturing material database and the second in OWL from their 2022 AM bench database. These use cases present the power of the AM-CDM for data representation, querying, and seamless data exchange. Our implementation experiences and some challenges are highlighted that can assist others in future adoptions of the AM-CDM for data integration and data exchange applications.

Keywords Additive manufacturing · Database · Data dictionary · Data exchange · Data structure

Introduction

Additive manufacturing (AM) promises to truly revolutionize large swathes of the manufacturing industry. Rather than traditional subtractive manufacturing in which a block of material, typically metal, is processed through various types of cutting operations to remove chunks of the block, additive manufacturing operates by repeatedly adding additional material and fusing that new material to the previous material. In this way, a new part is built up from raw material, usually resulting in significantly less waste.

✉ Yan Lu
yan.lu@nist.gov

Alexander Kuan
alexander.kuan@nist.gov

Kareem S. Aggour
aggour@ge.com

Shengyen Li
shengyen.li@nist.gov

Luke Mohr
lmohr@ewi.org

Alex Kitt
akitt@ewi.org

Hunter Macdonald
hunter.macdonald@mscsoftware.com

1 Division 734, National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA

2 General Electric Aerospace Research (GE), Niskayuna, NY, USA

3 EWI, Buffalo, NY, USA

4 Hexagon, Richmond, VA, USA

AM has been adopted on a small scale in many industries such as aerospace [1] and biomedical [2]. For example, GE has been credited with demonstrating the viability of additive manufacturing of mass-produced parts in the aerospace industry with its fuel nozzle [3]. However, AM is still a relatively nascent field and needs many enhancements to truly revolutionize the manufacturing industry on a massive scale. Many commercial companies, academic institutions, national laboratories, and standards organizations have been partnering for years to address some of the fundamental challenges and shortcomings related to AM to make it a viable technology.

One of the challenges, these organizations have routinely faced is how to effectively exchange data to facilitate their collaboration and streamline AM development for scale-up. Most data exchanges are performed on an ad hoc basis and rely on a priori agreements between the data senders and receivers on the data format, labels, and structure, for the receiver to be able to parse the data and do anything meaningful with it. This is the challenge of data interoperability in the context of the FAIR (Findable, Accessible, Interoperable, Reusable) guiding principles of scientific data management [4]. Even more troubling, it had become apparent that many within the AM community were using the same vocabulary but with different definitions for those terms.

Beyond the definitions of the vocabularies, the needs of the AM data models are addressed repeatedly. The data models contain rich sets of metadata with respect to the technical details in organized data structures. The data models enable the accessibility, traceability, and automation of the digital threads for the following analytical and numerical procedures [5–7]. They are also important to support the data sharing and exchange for different engineering applications, such as qualification and certification [8–10]. However, the lack of the collaborative data models creates the technical gaps for the development of the technology [6, 11]. A consensus agreed data model with clear definition of the vocabulary is critical to the data management for AM.

To alleviate these issues, a community of AM practitioners decided to form a committee to develop an additive manufacturing common data dictionary (CDD). The purpose of this AM-CDD was to develop an extensive dictionary of the common terms used within the AM community and give them a specific definition that everyone could rally behind. The AM-CDD further specified the data type of each term, provided examples of their units, and identified any existing standards that were available that could be used to set values for each term. The AM-CDD has been adopted as an ASTM standard—ASTM F3490-21 “Standard Practice for Additive Manufacturing—General Principles—Overview of Data Pedigree.”

To further ease the exchange of AM data among computer systems and enable interoperable data integration, a subcommittee of the AM-CDD working group formed two years ago to take the AM-CDD to the next logical step and define a structure around the CDD—by turning the data dictionary into an AM Common Data Model (AM-CDM) that additive manufacturing researchers and practitioners alike can use to model their data when they wish to exchange data across organizational boundaries.

The remainder of this paper is organized as follows. The benefits of a CDD and CDM are first elaborated in Sect. "Benefits of a CDD and CDM", including outlining how they can enable FAIR data within the AM community. The ASTM CDD standard and the AM-CDM that is under development are then both described in further detail in Sects. "AM Common Data Dictionary" and "Common Data Model", respectively. The paper then describes two use cases for the application of the AM-CDM to data curation for different purposes and using different technologies in Sect. "Use Cases: Using the AM-CDM to Curate Data from NIST AMMD and AM Bench 2022", followed by results and analysis in Sect. "Implementation Results/Analysis". The paper then concludes with a summary of this effort in Sect. "Conclusions".

Benefits of a CDD and CDM

The FAIR principles, an acronym for Findable, Accessible, Interoperable, and Reusable, represents a set of guiding principles for enhancing the management, sharing, and utilization of scientific data in today’s data-intensive research landscape [12]. These principles emphasize the importance of data discoverability, encouraging researchers and organizations to make their data *Findable* through comprehensive metadata, standardized identifiers, and clear naming conventions. *Accessibility* is the aspect of promoting open and easy access to data, either through public repositories or well-defined access protocols. By adhering to the principles of *Interoperability*, data can be seamlessly integrated and exchanged across different platforms and disciplines, enabling collaborative research and maximizing the value of data assets. Lastly, ensuring data are *Reusable* involves providing comprehensive documentation, context, and metadata, making it not only comprehensible but also usable by others, thus fostering scientific reproducibility and innovation.

CDDs are a key enabler of FAIR data in that they provide a common vocabulary that dramatically simplifies the findability and reusability of data. Without a common, shared vocabulary around data, users would not be able to search one another’s repositories and retrieve meaningful datasets from one another. APIs are less absolute if there is not a common understanding of what the values in a dataset represent. Common data dictionaries encourage collaboration between different teams and organizations by ensuring that data elements are understood consistently across the community.

While powerful, CDDs are limited in that they only allow humans to communicate more effectively. To truly enable FAIR data, we must go one step further and allow computing systems to communicate more effectively as well, and this is where a CDM comes into play. A CDM defines the structure and relationships around the vocabulary defined in the CDD. By defining the structure and relationships in the CDD, a CDM allows machines to exchange data that is expressed in the terminology of the CDD. Therefore, while the CDD is an important first step in allowing individuals within the AM community to communicate more effectively, it is the CDM that allows machines to communicate effectively in exchanging data and making it interoperable and reusable, thus making data fully FAIR.

The significance of CDDs and CDMs can be summarized as follows:

Standardization CDDs and CDMs establish standardized definitions, formats, and metadata for data elements. This consistency ensures that everyone who uses the data interprets and uses the data uniformly, reducing errors and misunderstandings.

Interoperability In a world where data are frequently shared across systems and organizations, CDDs and CDMs enable interoperability by ensuring that data exchanged between different systems adheres to a common set of standards, formats, and meanings. By adhering to a shared model, organizations can break down data silos and ensure compatibility among various data sources and systems.

Data Integration When organizations collect data from diverse sources, integrating these data becomes a complex task. CDDs and CDMs simplify data integration by providing a common language and structure, streamlining the data sharing process.

CDDs and CDMs are pivotal in today's data-driven landscape. They promote standardization, interoperability, and seamless data integration, making data more accessible and valuable within and across organizations. In essence, CDDs and CDMs are the linchpin of effective data sharing, promoting accuracy, consistency, and efficiency in exchanging data across organizational boundaries, and are pivotal to making data FAIR.

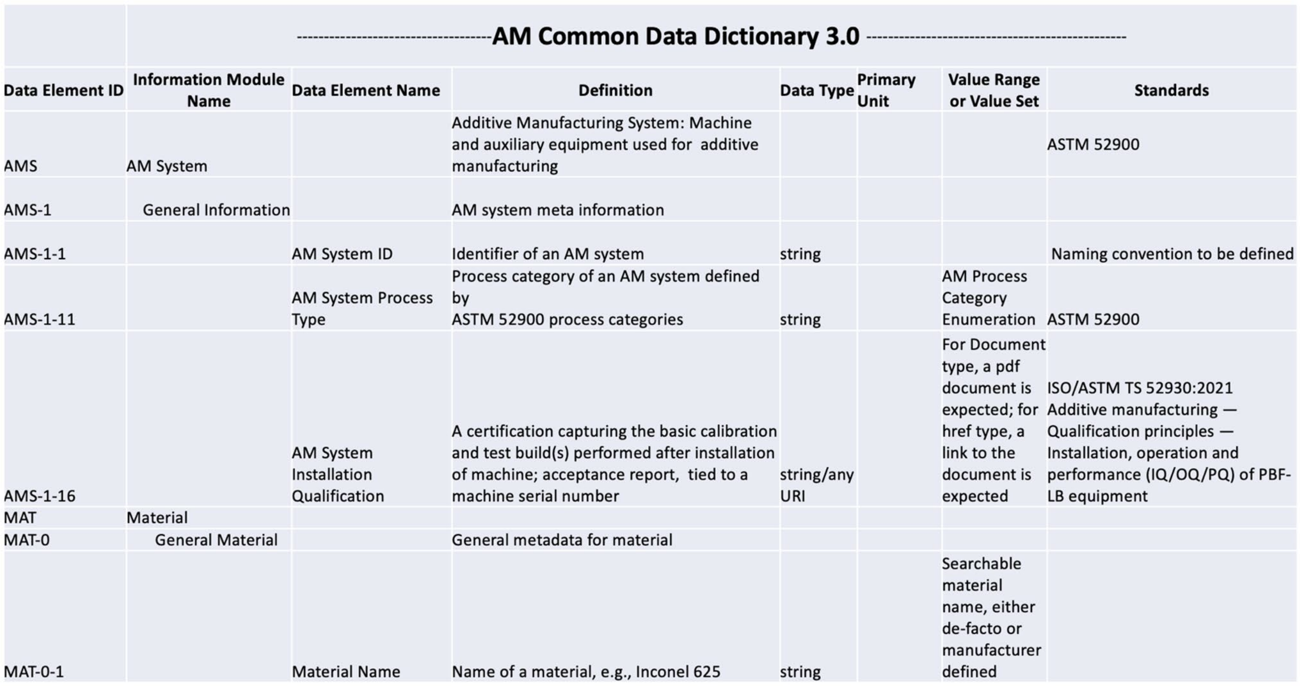
AM Common Data Dictionary

The AM-CDD has been developed to provide a consistent technical vocabulary of AM concepts and attributes for the community to effectively communicate and collaborate. The AM-CDD not only allows AM system developers to design or update a data store that meets business and process requirements using standard definitions of data elements, but also enables AM data sharing among organizations and personnel with legacy proprietary data systems using neutral definitions for essential AM data terms that can be mapped to proprietary data. Figure 1 shows an extract of the AM-CDD developed by a joint industry-government-academic working group.

The working version of the AM-CDD is defined in an online Excel spreadsheet that consists of several tabs to ease its development by a large, distributed team. The first tab provides an overview of the top fifteen AM concepts (also referred to as "buckets") and their relationships. These concepts are used to group AM data items into information modules, for example, AMS (AM system), BLD (build), Mat (material), PRC (process control), PRD (process data), PTD (part design), etc. The second tab captures the definitions of about 830 AM data items that were considered essential for the community to manage and exchange information.

The column titled "ID" represents the unique identifier for the data element corresponding to that row. The second column contains the names of the main or sub-buckets. The "Data Element Name" column contains the given name of the data element, followed by a column containing the definition of the data element. This is followed by a "Data Type" column, which contains the preferred kind of data that a particular data element should contain, e.g., string, integer, and float. The values in this column reference a list of data types in the "Data Type" tab of the spreadsheet. In the next column is the "Primary Unit," which represents the preferred unit that the data type should be reported as listed in the "Unit" tab. The "Value Range or Value Set" column refers to either the values contained within the "Value Set" tab or to a single foreign key or multiple foreign keys. The final main column of this tab is the "Standards" column, where any known applicable standard related to this data element is listed.

When the AM-CDD reached relative maturity, it was transferred to the ASTM F42.08 Data subcommittee for standardization. The subcommittee decided that the process-agnostic elements of the AM-CDD, representing a core of common attributes across all AM processes, should be included in the first AM data dictionary standard. After incorporating feedback from over one hundred domain experts over a period of a year and a half, ASTM F3490-21 "Standard Practice for Additive Manufacturing—General Principles—Overview of Data Pedigree" was officially adopted and released, which includes 395 general AM data items, and was defined as the first AM Common Data Dictionary standard.



-----------------------------------AM Common Data Dictionary 3.0 --

Data Element ID	Information Module Name	Data Element Name	Definition	Data Type	Primary Unit	Value Range or Value Set	Standards
AMS	AM System		Additive Manufacturing System: Machine and auxiliary equipment used for additive manufacturing				ASTM 52900
AMS-1	General Information		AM system meta information				
AMS-1-1		AM System ID	Identifier of an AM system	string			Naming convention to be defined
AMS-1-11		AM System Process Type	Process category of an AM system defined by ASTM 52900 process categories	string		AM Process Category Enumeration	ASTM 52900
AMS-1-16		AM System Installation Qualification	A certification capturing the basic calibration and test build(s) performed after installation of machine; acceptance report, tied to a machine serial number	string/any URI		For Document type, a pdf document is expected; for href type, a link to the document is expected	ISO/ASTM TS 52930:2021 Additive manufacturing — Qualification principles — Installation, operation and performance (IQ/OQ/PQ) of PBF-LB equipment
MAT	Material						
MAT-0	General Material		General metadata for material				
MAT-0-1		Material Name	Name of a material, e.g., Inconel 625	string		Searchable material name, either de-facto or manufacturer defined	

Fig. 1 AM common data dictionary extract. Reprinted with permission from [12]

Common Data Model

The AM-CDD is incredibly important in that it standardizes the vocabulary of the additive manufacturing community. However, the AM-CDD by itself only allows living, breathing members of the community to communicate more effectively. To achieve truly FAIR AM data, a common data model and common data exchange formats are required to be established and adopted to enable the electronic exchange of AM data. Common Data Models are critically important in making the AM-CDD practical because they put a structure around the AM-CDD so that computing systems can begin to use the same vocabulary to make data Findable, Accessible, Interoperable, and Reusable.

By defining a formal class structure and defining the relationships between the terms within the AM-CDD, the precise linking between information fragments becomes apparent in a computable model. This computable model can be used by software systems to map specific tables or fields within a data store to neutral properties defined by the model. This makes those data stores linkable and allows distributed data to become findable using the vocabulary defined in the AM-CDD even if the underlying data stores do not store their data using the vocabulary specified in the AM-CDD.

This concept is shown in Fig. 2, in which a user can explore and select a set of attributes of interest in terms defined within the AM-CDM. This query can then be pushed to one or more underlying data storage systems across many different potential AM collaborators. Each of these collaborators may have their own unique underlying storage technologies with their own data schemas and structures, they do not have to adopt the AM-CDM as their internal data representation to make their data FAIR. Collaborators must, however, map their data to the AM-CDM so that when a query is sent to their data system, their infrastructure can automatically parse the AM-CDM-based query, translate it into a query executable against their internal data structure, retrieve whatever data are shareable that matches the search criteria, and then return the data in the structure of the AM-CDM.

Figure 2 mentions four specific systems (from GE, NIST, AFRL, and Hexagon), and a generic fifth "System X" to highlight the fact that this approach is flexible and extensible to any participant that would like to make their additive manufacturing data FAIR. These four specific systems are highlighted as examples because members of these organizations have been partnering on the development of the AM-CDD, AM-CDM, and AM-CDEF and have (e.g., through AFRL's CAMDEN effort) and/or are actively developing proofs-of-concept to demonstrate that these kinds of data exchanges can work.

Figure 2 showcases a vision of a common data model being used to enable access to multiple source systems, each with their own internal data structures. A user can specify attributes of interest within some query system (as

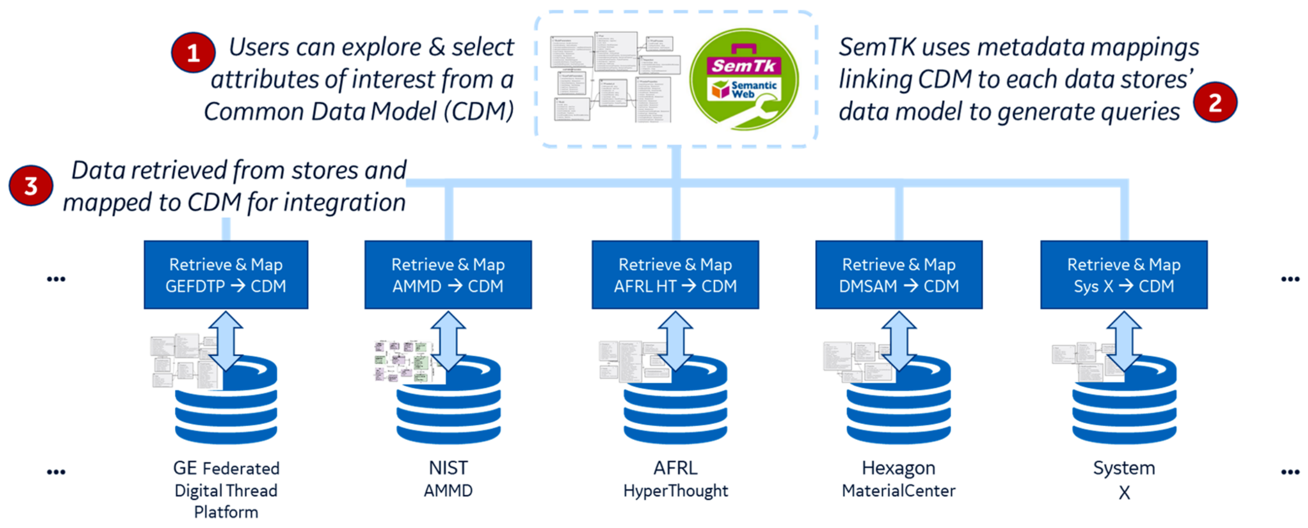

1 Users can explore & select attributes of interest from a Common Data Model (CDM)
SemTk Semantic Web
SemTK uses metadata mappings linking CDM to each data stores' data model to generate queries 2
3 Data retrieved from stores and mapped to CDM for integration
Retrieve & Map GEFDTP → CDM
Retrieve & Map AMMD → CDM
Retrieve & Map AFRL HT → CDM
Retrieve & Map DMSAM → CDM
Retrieve & Map Sys X → CDM
GE Federated Digital Thread Platform
NIST AMMD
AFRL HyperThought
Hexagon MaterialCenter
System X

Fig. 2 Vision of a common data model with multiple source systems. Reprinted with permission from [12]

represented within the dotted box at the top of Fig. 2), potentially using a graphical user interface to visually explore classes within the AM-CDM to select specific attributes. An open-source tool such as the Semantics Toolkit (SemTK) [13], could transform the AM-CDM-based request into one or more queries against the underlying source systems, which would in turn run the queries, retrieve their local data, and map it back into a format that conforms to the AM-CDM so that the data from the many systems could be seamlessly merged.

Common Data Exchange Formats

Local data models are generally designed based on different design philosophies for unique purposes. The integration of data across different infrastructures requires the identification of desired attributes and then the retrieval and physical merging of data between multiple distinct local data models. The design and development of common data exchange formats is an effort to eliminate the technical barriers for the transmission and integration of data across siloed infrastructures.

An intelligent data exchanger should be capable of fusing datasets from different sources, such as SemTK [13] as shown in Fig. 2. Integration using APIs and open-source software libraries can satisfy the needs at the executive level of data federation. The major challenge is that the local data models may be implemented using different computer languages and different file formats for data curation, which requires additional steps to compare and align the information. A common data exchange format is needed to translate the ontologies and taxonomy into a widely accepted format and align relevant information from multiple sources.

By developing common data exchange formats that align directly to the AM-CDM, we eliminate the need to design and develop custom data exchange formats for every application and every scenario in which multiple parties wish to exchange data. Whenever two or more parties wish to exchange data, they simply need to agree on what data they wish to exchange, and then can instantiate a data file or files that match the AM-CDM component(s) that align to the data they wish to exchange, and then in an instant they will have a common data exchange format file that will meet their needs without having to go through a standards body and months or years of approvals. In this way, the AM-CDM drives not just standardization in the way data is modeled and queried across systems enabling findability and accessibility, but also enables rapid transmission and exchange of data, facilitating data interoperability and reuse.

AM-CDM Design

The AM-CDM has been divided into six logical modules, helping to facilitate its parallel development. These six modules are: base, material, system, process, build, and TIC (test-inspection-characterization). Some of the core classes and their connections from these modules are shown in Fig. 3.

The AM common data model is available to the public in a GitHub repository located at: https://github.com/kaggour/AM-CDM. It has been developed in the Semantic Application Design Language (SADL), a formal, English-like language and Eclipse plugin developed to simplify the development of semantic models by non-semantic modeling experts [14]. SADL allows experts in broad domains, such as additive manufacturing and materials science, to read and write SADL without having to become experts in semantic

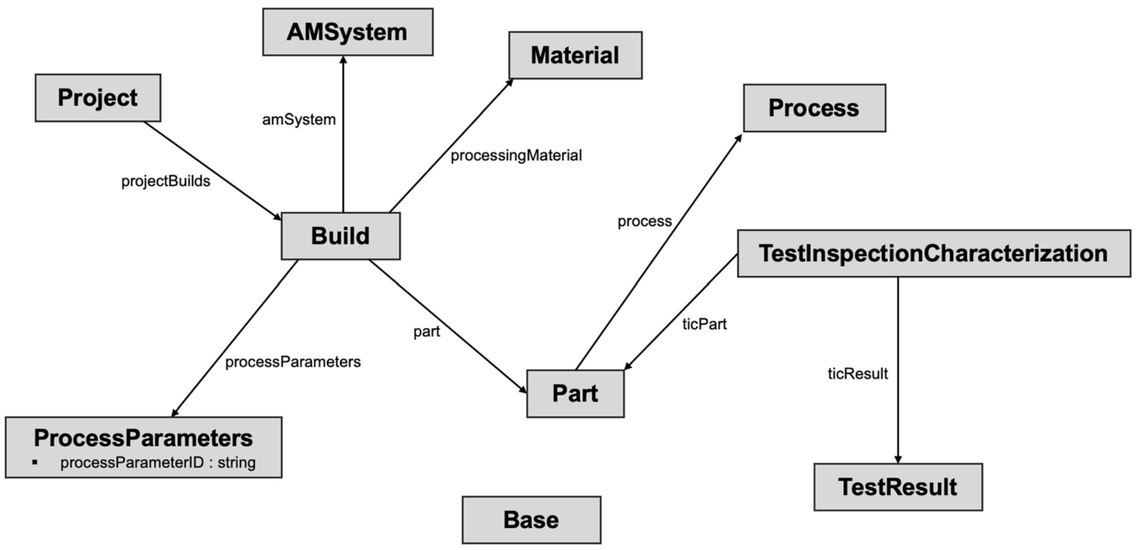


Fig. 3 Examples of core classes in AM-CDM and their relationships. Adapted with permission from [12]

modeling. One powerful feature of SADL is it can autogenerate OWL (the web ontology language) and so both the SADL and OWL versions of the CDM are available in the Git repository. While we have used a semantic language editor to develop the CDM, the CDM is not an ontology, it is a data model meant to define the structure and relationships of a set of terms that have been defined elsewhere (the AM-CDD). Each of the six modules are described below.

Base Module

The base module consists of over a dozen foundational classes such as person, organization, qualification, measurement (the primary class used to store data values, including the numerical value and unit, among other meta information), and more. Each of these foundational classes contain a variety of attributes specific to each class. For example, qualification includes qualificationType, qualificationLevel, and qualifyingOrganization attributes, and Person includes personID, personFirstName, and personQualification, to name a few. Included with the class attribute definitions are their data type and cardinality. In the case of the Person class, each person has a single first name of primitive data type string but can have multiple qualifications of type qualification. An example of the person and qualification base classes are shown in Fig. 4. Note that some attributes are defined as single-valued ("with a single value of type…"), and other attributes are defined as potentially multi-valued ("with values of type…").

In Fig. 4, the person and qualification base classes are written in the semantic application design language, in which each class has multiple attributes of both primitive (e.g., string, double, float) and complex data types associated, which link different classes together.

Material Module

Materials are prepared in different shapes, dimensions, phases in thermodynamics, and chemical properties for different AM technologies. It should be addressed that the material mentioned here is different from the material as a part for the assembly of a system, which requires different prospections for the location-specific properties, which is beyond the scope of this work. Being more specific to metal AM, we include a variety of attributes in the material module such as (1) the alloy composition, and (2) many intrinsic properties, such as the specific heat and thermal conductivity, as well as (3) extrinsic properties with some associated variability of materials, such as size distribution of the powder particles and tensile properties as shown in Fig. 5, as these are critical to the design of the AM build strategy and to assess the quality of the products.

Also, important to processing history is the initial status of the material, which is vital to all kinds of engineering projects and quality assurance. The information about the manufacturer, production batch, and storage environment help to clarify the root cause of potential sources of variations in an enormous space. The number of reuses/recycles is also a part of the material history affecting the variability of the material status and is captured as attributes of the Material class.

```
Person (note "Individuals and their primary credentials and contact information.") is a class,
    described by personID (note "Unique identifier of the person.") with a single value of type string,
    described by personFirstName (note "First name of individual performing task.") with a single value of type string,
    described by personLastName (note "Last name of individual performing task.") with a single value of type string,
    //described by personRole (note "Role of individual performing task.") with a single value of type string,
    described by personEmail (note "The email for the individual.") with a single value of type string,
    described by personPhoneNumber (note "The phone number for the individual.") with values of type PhoneNumber,
    described by personQualification (note "Qualifications of person.") with values of type PersonQualification,
    described by personOrganization (note "Links person to an organization") with a single value of type Organization.

Qualification (note "Details of qualification/certification (machine specific operator certificate, UL, ASTM).") is a
    class,
    described by qualificationType (note "The type of qualification/certification.") with a single value of type
    string,
    described by qualificationLevel (note "The level of qualification/certification.") with a single value of type
    string,
         described by qualifyingOrganization (note "The organization from which the qualification or certification
originates.") with a single value of type Organization.
```

Fig. 4 Person and qualification base classes written in the semantic application design language. Reprinted with permission from [12]

```
TensileProperties is a type of ExtrinsicMaterialProperties,
     described by tensileTestStrainRate with a single value of type Measurement,
     described by tensileTestDisplacementRate with a single value of type Measurement,
     described by tensileTestControlMode with a single value of type string,
     described by initialStrainRate with a single value of type Measurement,
     described by initialLoadingRate with a single value of type Measurement,
     described by finalStrainRate with a single value of type Measurement,
     described by finalLoadingRate with a single value of type Measurement,
     described by yieldStrengthZeroPoint2Percent with a single value of type Measurement,
     described by yieldStrengthZeroPointZero2Percent with a single value of type Measurement,
     described by ultimateTensileStrength with a single value of type Measurement,
     described by elongation with a single value of type Measurement,
     described by percentReductionInArea with a single value of type Measurement,
     described by youngsModulus with a single value of type Measurement,
     described by maximumLoad with a single value of type Measurement.
```

Fig. 5 TensileProperties class in material module written in the semantic application design language

System Module

The system module of the AM-CDM describes the physical equipment and the software associated with additive manufacturing processes, including the 3D printers themselves as well as non-AM auxiliary equipment. All physical, equipment-based classes extend the root "System" class, which includes several generic attributes such as systemID, systemName, systemManufacturer, and systemModel.

Additionally, the system module includes both AMSystem classes and NonAMSystem classes. Many modality-specific classes further extend AMSystem, including LaserPowderBedFusionSystem, DirectedEnergyDepositionSystem, and BinderJetSystem, to name a few. The System class also includes meta information about the machine's capabilities, classes, and attributes about the software installed on the machines, and information about the maintenance, configuration, and calibration of both the hardware and software.

Overall, the system module of the CDM is used to model information about the hardware and software used in the end-to-end additive manufacturing process.

Build Module

The build module models all the information generated during a single AM process cycle during which one or more components are "built up" in layers inside the process chamber of the additive manufacturing system (ASTM 52900). The main Build class acts as a central reference point for all data related to a build, as shown in Fig. 6. The definition of the Build class not only has all the metadata attributes that describe a build, such as buildID, buildType, and buildTime, but also contains all the attributes linking to other modules, e.g., feedstockMaterial and AMSystem. Other attributes, including part, buildPlatform, buildParameters, amInsituData, buildSimulation, and buildSoftware, are defined by the classes in the Build module.

```
Build (note “A build is an action to make one or more objects in a single action by an additive manufacturing
system. A Build record encompasses all of the data for the build cycle; and single process cycle in which one or
more components are built up in layers in the process chamber of the additive manufacturing system.”) is a type of
ProcessStep,
    described by buildID (note "An identifier for an AM build.") with a single value of type string,
    described by buildType (note "Description of the primary purpose of the Build (end use component, specimens,
    research, etc.).") with a single value of type string,
    described by buildSimulation (note "Reference to Simulation records associated with this build.") with a
    single value of type BuildSimulation,
    described by feedstockMaterial (note "Material consumed during build.") with a single value of type Material,
    described by quantityOfFeedstockMaterialUsed (note "Mass of material used in solidified parts.") with a single
    value of type Measurement,
    described by processingMaterial (note "Ancillary material used for build, e.g., environmental gas.") with a
    single value of type Material,
    described by buildPlatform with a single value of type BuildPlatform,
    described by part (note "Parts built from this build. One build can have multiple built parts.") with values
    of type Part,
    described by buildPlan (note "Document that describes the build specifc design intent and requirements. This
    may include plate layout, witness coupons, object quantity, position/orientation.") with values of type
    Document,
    described by buildSoftware (note "Software used in build.") with values of type Software,
    described by buildOrganization (note "Production organization for this build.") with a single value of type
    Organization,
    described by buildLayout with a single value of type Document,
    described by buildJobFile with a single value of type Document,
    described by amSystem (note "AM system used for this build.") with a single value of type AMSystem,
    described by amInSituData (note "Advanced AM in-process monitoring BIG data, is differentiated from
    processData.") with values of type AMInsituData,
    described by buildDuration (note "Recorded duration of build cycle.") with a single value of type Measurement,
    described by buildLocation (note "Place of an AM process.") with a single value of type Address,
    described by buildCooldownDuration (note "Time from build completion to cooling to 100 degrees C.") with a
    single value of type Measurement.
```

Fig. 6 Build class in build module written in the semantic application design language. Reprinted with permission from [12]

Part refers to instances of 3D designs in the as-built state made in this build cycle. One Build can have multiple build Parts of the same partDesign. A partDesign can be realized by many Builds. In this sense, PartDesign can be defined independent of Build. However, for this data model, PartDesign is not used by any other modules, hence the class is defined in the Build module. buildParameters extends processParameters which can have attributes defined by multiple process parameters or just a build command file. The amInsituData extends the datasetMetadata class defined in the Base module with additional attributes associated with the in situ monitoring device, the reference framework of the device, the configurations, and the data acquisition information. BuildSimulation and SynthesizedData classes define the metadata that describe a simulation of the build, and the data analysis results for the build.

Process Module

The process module defines all the classes that are necessary to describe an additive manufacturing process and the process sequence associated with a part or specimen. The ProcessStep class refers to a manufacturing activity performed as a component of an ordered sequence, which include preprocessing, build, post-processing, and (destructive or non-destructive) test, inspection, and/or characterization operations. A base class of ProcessStep is defined as shown in Fig. 7.

Important attributes of ProcessStep include processParameters, processControlPlan, and processData. The first two attributes define the controls of a manufacturing process parametrically and with a document, respectively. processData captures the information about the measured data, or derived outputs from that data, obtained from the AM system and in situ monitoring equipment during a build process. Type-specific information about a manufacturing process is defined in the extensions of the ProcessStep class, such as Build, HeatTreatment, TestInspectionCharacterization, etc.

Test-Inspection-Characterization Module

The test-inspection-characterization (TIC) module can be thought of as an information space that includes measurement methods, structural features, and material properties, all of which are based on the materials data and the processing history. AM projects need TIC for designing and assessing the building strategies, as well as the post-building treatments. Each material system, measurement method, and targeted application have unique domain knowledge and requirements. Developing a thorough ontology to cover all possible aspects can be very challenging and so we focus on a high-level structure to convey the concepts of the model development for TIC.

```
ProcessStep (note "Event performed as part of a sequence. May include pre-processing, build, post-processing, and
(destructive or non-destructuve) test, inspection, and/or characterization operations.") is a type of Process,
    described by sequenceNumber (note "Sequence identifier of processing steps in chronological order.") is a type
of ProcessStep,
    described by processID (note "Unique identifier for a processStep.") with a single value of type string,
    described by processStartTime (note "Time when process step starts.") with a single value of type dateTime,
    described by processEndTime (note "Time when process step ends.") with a single value of type dateTime,
    described by agentOrganization with a single value of type Organization,
    described by agentPersonnel with values of type Person,
    described by processMachine with a single value of type System,
    described by processData with a single value of type ProcessData,
    described by processParameters with a single value of type ProcessParameters,
    described by processStandard (note "Standards used to guide a process, e.g., AMS2774.") with values of type
Document,
    described by processControlPlan (note "Document that describes the methods taken to ensure the quality control
of critical inputs to deliver outputs that meet requirements.") with a single value of type Document.
```

Fig. 7 ProcessStep class from the process module. Reprinted with permission from [12]

Starting from the top level in Fig. 8, the metadata for TestInspectionCharacterization includes testing facility, testing conditions, operator, and environmental conditions. Testing facility refers to the hardware model, software version, and calibration schedule. Testing conditions indicate the shape and size of the coupon and testing variables such as strain rate and temperature for tensile tests. Operator and environmental conditions may cause variability in the outcomes, and hence the need to capture such metadata. Because the results of TIC highly depend on the status of raw material

```
TestInspectionCharacterization (note "Any test, inspection, or characterization performed on any part or material
specimen from any stage of the additive manufacturing fabrication process (excluding in-process monitoring).") is
a type of ProcessStep,
    described by ticID (note "An identifier of the test, inspection or characterization type.") with a single
value of type string,
    described by ticName (note "A short description of the test, inspection, or characterization.") with a single
value of type string,
    described by ticType (note "Type of test/inspection/characterization, such as tensile test, fatigue test,
etc.") with a single value of type TICType,
    described by ticStandard (note "An identifier of the corresponding standard used.") with a single value of
type string,
    described by ticMaterialSpecimen with a single value of type MaterialSpecimen,
    described by ticPartExtract with a single value of type PartExtract,
    described by ticPart with a single value of type Part,
    described by ticStartTime (note "Start time and date of the test, inspection, or characterization.") with a
single value of type dateTime,
    described by ticEndTime (note "End time and date of the test, inspection, or characterization.") with a single
value of type dateTime,
    described by ticLocation (note "The physical location where the test was conducted.") with a single value of
type Address,
    described by ticNotes (note "Itemized descriptions of observations relating to the individual test,
inspection, or characterization of an individual specimen.") with values of type string,
    described by ticOperator (note "Identifier of the operator who facilitated the test, inspection, or
characterization linking to Personnel ID.") with a single value of type Operator,
    described by ticPointOfContact (note "The name of the point of contact for the task being performed, if not
the person performing the task.") with a single value of type PointOfContact,
    described by ticVendor (note "Identifier of Vendor/Supplier/Contractor who physically performed the test,
inspection, or characterization linking to Organization ID.") with a single value of type TICVendor,
    described by ticEquipment (note "Identifier of Non-AM Equipment used during the test, inspection, or
characterization.") with a single value of type NonAMSystem,
    described by ticSoftware (note "Identifier of any Software that was used during the test, inspection, or
characterization.") with a single value of type Software,
    described by ticProcedure (note "The procedure used if it is not from an existing standard.") with a single
value of type Document,
    described by ticDestructureVsNonDestructive (note "Indication of whether the test, inspection, or
characterization irreversibly changed the nature of the specimen.") with a single value of type TestType,
    described by ticDuration (note "The time that the Test took (Test only).") with a single value of type
Measurement,
    described by ticEnvironment (note "The atmospheric conditions of the test environment.") with a single value
of type TICEnvironment,
    described by ticResult with a single value of type TestResult.
```

Fig. 8 TestInspectionCharacterization class from the TIC module. Reprinted with permission from [12]

and processing history, appropriate links or a handler system is also required to create a comprehensive dataset.

Use Cases: Using the AM-CDM to Curate Data from NIST AMMD and AM Bench 2022

The AM-CDM development is sufficiently mature for stress testing. To demonstrate the use of the AM-CDM for database development, this work adopts two public datasets—the AM Bench 2022 dataset [15–17] and the additive manufacturing materials database (AMMD) [18]. These datasets were generated from eight distinct projects that used different AM approaches for different program objectives. These datasets provide a rich set of data and metadata for testing the AM-CDM, containing 1067 measurement results from 37 AM builds using 6 different materials.

Figure 9 provides an overview of the datasets used in this work. The data have been separated into five categories: source of the *Database*, information on *Project* management, feedstock *Material* data, controls and environmental conditions of AM *Build*, and *Measurement* results. Each category contains a certain number of nodes, in which each node represents a collection of information. For example, a *Build* node represents a set of the metadata from a specific AM build process that may include the information about the AM machine, environmental conditions, layer thickness, laser scanning strategy, and so on. The *Measurement* nodes connect to *Build* nodes to save the results from structure characterizations and mechanical tests. The former includes microstructure and part features such as grain size and surface roughness, and the latter contain results such as tensile and fatigue properties. This case study uses this information to evaluate the implementation of the current AM-CDM using an XML-based document database and an ontology-based graph database to assess the strengths and weaknesses and compare the merits of different approaches.

Model Implementation: XML and Graph-based (OWL)

The AM-CDM focuses on a modular architecture in which classes can form relationship links to any other class within the model. The current AM-CDM has no main class or classes defined as the root of a tree structure, so finding the core starting point of the hierarchy is at the user's discretion. The definition of the classes enables flexibility and extensibility of the data architecture, and class-to-class relationships are not limited by the structure of a rigid class hierarchy. Additionally, making the design of the AM-CDM relationship-focused enables many systems with different data storage approaches to utilize the common data model for AM data and metadata modeling and storage.

Ultimately, the goal of the proposed database with the AM-CDM is to allow cross-platform data integration, therefore, different users with different systems should be encouraged to adopt the model for their own use cases, such as the document-based database and graph database presented in this study. By continuing to implement the CDM for database testing across diverse use cases, the team can perform sufficient analysis and generate constructive feedback to ensure the AM-CDM is sufficiently robust to meet the objective of enabling data integration and exchange across a wide variety of systems and applications.

XML Document Database

The proposed XML schema includes six classes of the metadata types for saving a set of data in one document with a simple hierarchy. The hierarchy is shown in Fig. 10. The top

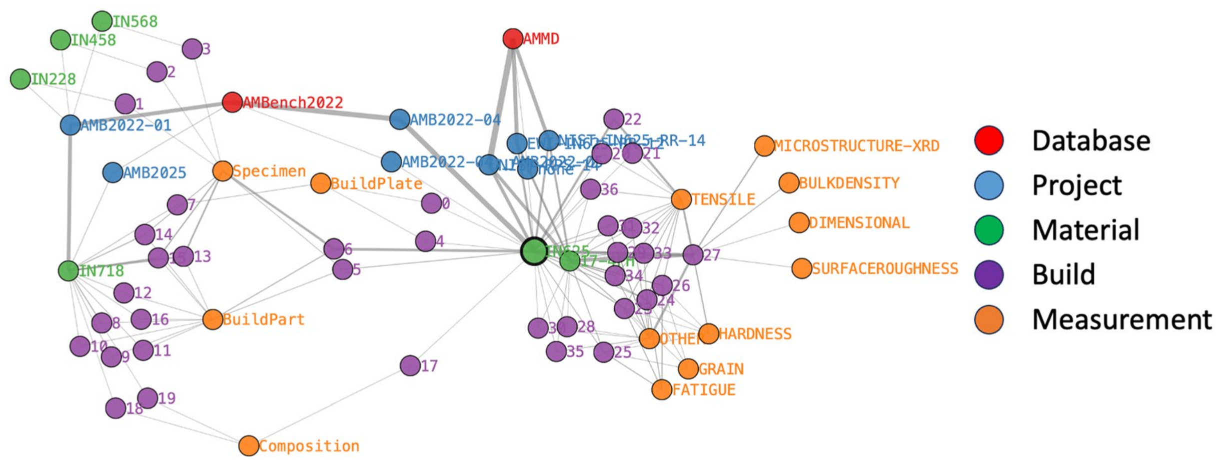


Fig. 9 Diagram of the publicly available datasets being merged in this POC

layer adopts the Project class from the AM-CDM build module to save the project details, such as the project ownership, point of contact, project plan, etc. Among these metadata, project ID is the key identifier to label the data pedigree that connects all the records to address the findability of the project deliverables.

AM Bench 2022 reports rich measurement results. To make these results findable, the major component of the proposed XML database is implemented from the AM-CDM TIC model called the Measurement class. As shown in Fig. 10, each Project may include several types and numbers of measurement results that include not only the procedure and results of the measurements but also the specimen history before testing. The specimenMaterialSource class in the Measurement class is designed to save the feedstock information. It uses the AM-CDM Material model to save the material chemistry, stock and batch number, material geometry, etc. The material has been processed before the measurements and the specimenSamplingMethod is used to record the machine setups and processing history, including the AM build process and post-build treatments. Among all the data types in the specimenSamplingMethod class, the processMachine and AMBuild classes call the System and BuildType from AM-CDM to save the AM machine, build plan, machine software, operation parameters, etc. These metadata are not limited to a specific AM process and can be applied to non-AM processes as well.

Besides project ID, this database assigns identifiers to each measurement result, process, and material to create a project pedigree. As shown in Fig. 10, one materialID is assigned to a specimenMaterialSource record. Because the same batch of material can be used for different processes followed by different Measurements, the specimenMaterialSource record is reusable by calling the materialID. A similar concept is applied to the design of specimenSamplingMethod; however, a complete process can include as many as processing steps in a sequence of processIDs. To apply this database in practice, one must carefully use the identifier system to assign unique projectIDs and other identifiers to construct a data pedigree.

Graph Database

The NIST AMMD is a legacy data store developed on top of MongoDB, a NoSQL technology. Currently an ongoing research effort is to convert the NIST AMMD to a graph database that allows for easier data extension, distributed data curation, and automated AM big data ingestion through data streaming. Some of the AMMD datasets relate to experimental setups, such as the *Three-Dimensional Scan Strategies* study [19]. An open-source technology from Idaho National Lab named Deep-Lynx [20] is being adopted for this project. Deep-Lynx needs a metadata structure to define all the node types and their relationships. The AM-CDM is a perfect candidate, however, with some necessary customization. After the specific tunings needed to be compatible with Deep-Lynx, the OWL version of AM-CDM can be directly loaded into Deep-Lynx as a baseline to display the node types and their relationships. When mapping metadata into node and edge relationships, the AM-CDM assists

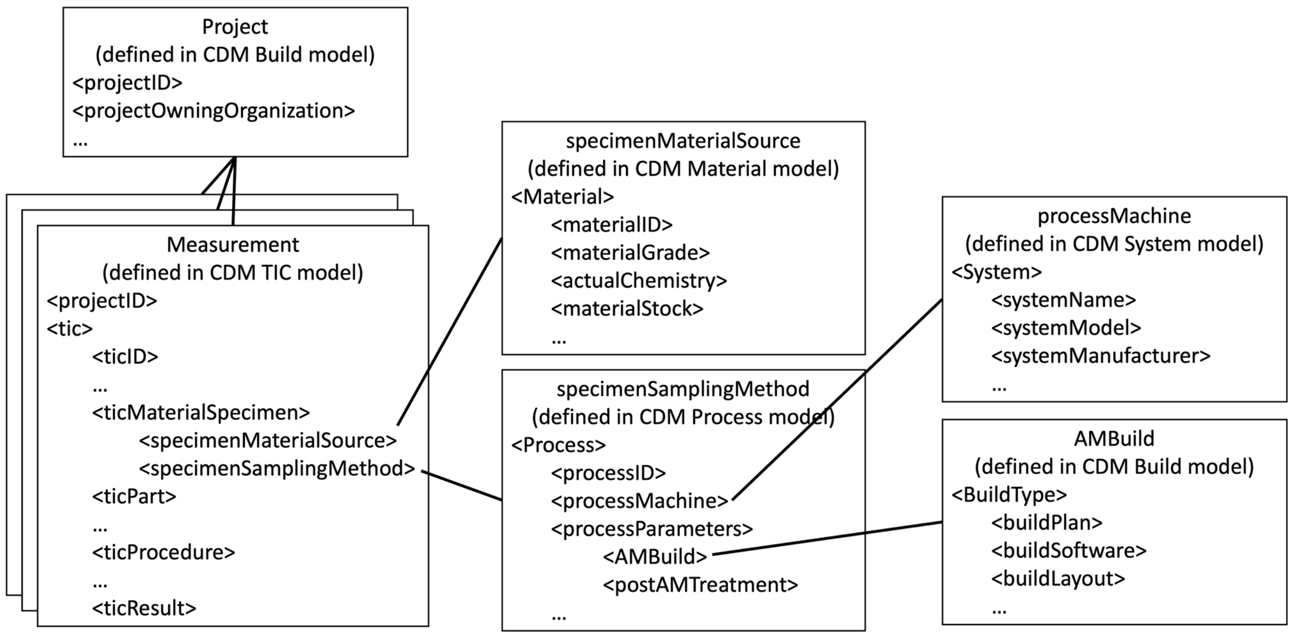


Fig. 10 Selected metadata presented in the six classes of the proposed hierarchy

in enforcing validation guidelines by using the properties/attributes defined within each class.

NIST AMMD employs an XML schema consisting of similar classes defined in the AM-CDM; however, the AM-CDM defines the relationships among the classes more explicitly, with greater visualization and comprehension of the entire dataset in the graph database. For instance, the Project class of the AM-CDM stores crucial metadata about a certain AM project; however, the graph database does not require the same project ID defined in a Project node and a Build node to form a relationship link between them, as displayed in the graph view. Connections between class nodes and metadata can be defined by linking certain metadata values stored in each separate node with predetermined conditional statements. The option to link by metadata values allows flexibility when mapping import payloads.

The difficulty of using the AMMD dataset is the distinct relationships defined in the XML schema compared to a graphical node and edge structure. Fortunately, the AM-CDM provides some form of a hierarchy architecture envisioned by the user. Just like the structure of the XML dataset, the core classes, such as Project and Build, stand out as the start or focal point of the dataset in the graphical view, but instead of encapsulating classes into subclasses, the visual representation of each node in a cluster appears as individual and determining edge relationships is focused on the correlation of metadata attributes rather than relying on a predetermined nested tree structure of parent and child classes. Figure 11 depicts a sample of the general AM-CDM main classes as node relationships in a graph view.

Note that the arrows in Fig. 11 display a defined relationship between two different classes. The arrow direction points from the class containing the relationship attribute and points to another specific node of the desired class type.

Implementation Results/Analysis

AM metadata is expansive, hence the difficulty of developing an ideal data model for public use. To address this challenge, the AM-CDM is being developed through a joint effort composed of industry, government, and academic stakeholders with diverse areas of expertise. However, the adoption and implementation of the AM-CDM highly depends on the database technology and the user objectives. For example, the defined relationships between two classes of the AM-CDM works well to structure metadata in XML document databases, but the same component created mapping issues within the Deep-Lynx graph database. The effort in testing AM-CDM with the use cases of various database technologies will help enable future cross-platform interoperability with multiple database types.

XML Implementation Analysis

The proposed XML-based document database is constructed using the classes and properties from the AM-CDM to archive and share the AM Bench data with R&D communities. Because the AM-CDM is documented using SADL, this development process translates the AM-CDM

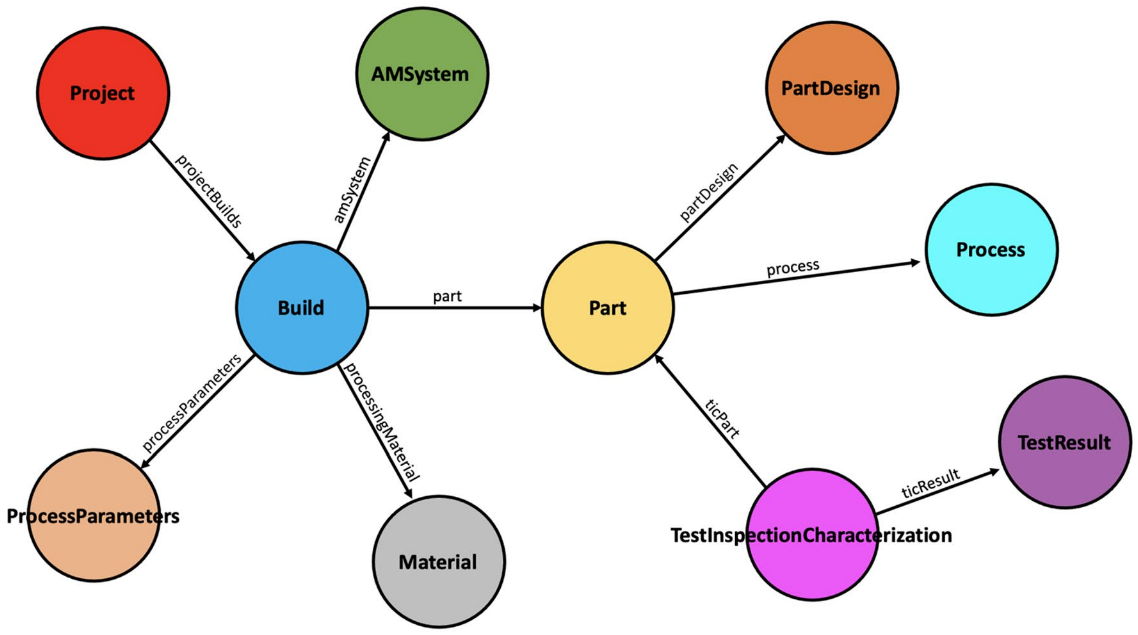


Fig. 11 AM-CDM graph database main class relationships

definitions to XML and completes the proposed structure in XSD format. The XSD schema enables an easy data entry of AM Bench 2022 datasets and pedigree information into a document database based on MongoDB. Similar to many data warehouses, the document database requests the key metadata from users to ensure the consistency and completeness of the data. It should be highlighted that because the XML schema based on the AM-CDM is well-structured, the database clearly is capable of capturing complicated datasets from AM. Also, the AM-CDM-based schema offers the scalability and reusability that are well-designed in the common data model for the advancement of AM technology. The AM-CDM provides fundamental definitions of data types that enable an ecosystem to reuse data types and lower the technical barrier for the scaling of the data model. The application of the AM-CDM to curating AM Bench data into an XML-based document has proven the hypothesis.

Graph Database Implementation Analysis

To perform graph database and the associated baseline metadata gap analysis for AMMD data migration, the graph database uses the same class properties from the AM-CDM documented in SADL to assemble, upload, and parse a structured OWL ontology file into the graph database system. The AMMD XML schema is mapped into the node and edge relationships defined by the object properties and data properties of the OWL model. As for the selected graph database, we use Deep-Lynx to automate the metadata mapping processes using the appointed AM-CDM structure.

In comparison to XML-based databases where classes are categorized and linked by ID values in separated documents, the current AM-CDM structure for a graph database focuses more on capturing multiple classes intertwined by direct relationships consolidating metadata into one graph node network. Graph databases utilize a graph composed of node classes containing metadata properties/attributes and edge relationships connecting each individual node. Instead of grouping main classes into separate tree hierarchies, relationships defined in the AM-CDM point directly to and from each class consolidating formation of all metadata for quick data queries. By the design of graph database systems, examining the graph view and searching for specific metadata can be more efficient and allows users to visualize relationships across all data within the graph. When querying data, the user may choose to search for specific node data, and the user can potentially find the exact desired node and all relationships between other nodes depending on the input of the search parameters. This makes graph databases highly beneficial for AM research and data analytics.

What holds graph databases back is the delicate nature of the graph view. Due to the relationship-heavy data architecture of a graph database, a highly developed data model must be implemented within the graph data system. Without a highly developed data model, data imported into the database may be lost or incorrectly mapped, which results in a higher risk of more transformation processing issues when converting data into node and edge relationships for the graph view. This can be very consequential as it may break the graph due to missing relationships between two or several significant node entities. The AM-CDM provides a baseline ontology for the graph database population to import the AMMD data, which greatly reduces the data migration time.

AM-CDM Implementation Comparison Between XML and Graph Databases

To compare the merits of the document and graph database implementations of AM-CDM, we list six aspects, shown in Fig. 12, summarizing the experiences from engineering developments to user experience. These qualitative comparisons are based on the XML and OWL databases running on CDCS and Deep-Lynx, respectively.

(1) *Easy implementation* refers to the technical difficulties while developing databases using AM-CDM. The development of an XML-based schema only needs to construct a data hierarchy using the data types from the AM-CDM; however, it needs additional effort to examine the linkages between data nodes for a graph database. To this point, the development of an XML-based schema is relatively easy.
(2) For *model validation*, XML only requires a single step of a software check. However, *model validation* for a graph database may need iterative, manual examination to confirm the linkages between data nodes.
(3) Manual data entry is a process highly dependent on the software and its user interface. With version 1.3.2 of Deep-Lynx, users need to select the appropriate classes and attributes, which requires the user to be familiar with the definitions of the entire AM-CDM for efficient data entry. For XML, the CDCS presents the XML schema in a tree structure that assists in better identifying the location of the metadata.
(4) In this work, *data completeness* means the level of required metadata is entered. Since both databases are AM-CDM based, the two instances perform equally in this item.
(5) The graph database provides efficient *pedigree querying*. In contrast, XML uses identifiers to connect the models that need additional effort to retrieve a set of relevant data.
(6) Similar to a data query, being capable of *visualization* of a complicated dataset is one of the merits of the

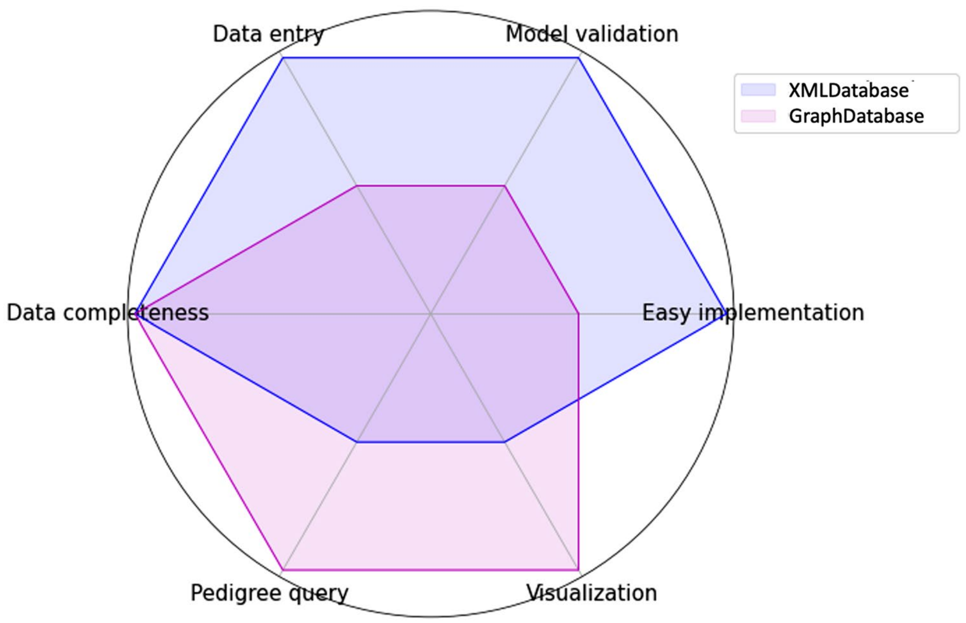


Fig. 12 Notional comparison of the XML and graph databases

graph database, but the XML databases require more effort to retrieve and reorganize the metadata.

Conclusions

This work introduces the AM-CDD and AM-CDM for the development of a common data vocabulary and data model to facilitate data exchange among AM data practitioners and systems for additive manufacturing projects. The AM-CDD defines the standard vocabulary and the AM-CDM creates data structures to accommodate the data and metadata including the information about project management, AM build, machine, process, and test-inspection-characterization.

Two use cases demonstrate the design and implementation of the AM-CDM into a document database (NoSQL) and a graph database to store the public datasets from the NIST AM Bench and AMMD databases. The case studies show the comprehensiveness and structural power of the AM-CDM in representing various AM information and data object types and relationships. The use cases have demonstrated the maturity of the AM-CDM for additive manufacturing data standardization. Each use case can also assist in the future developments of the standards for AM applications by providing insights on the best practices in the use of the AM-CDM, as well as feedback to the team on strengths and weaknesses of the AM-CDM.

Acknowledgements The authors appreciate the discussions and feedback from Peter J. Coutts, Dr. Milos Drobnjakovic, Dr. William “Pilgrim” Frazier, Duncan W. Gibbons, Dr. Joy Gockel, Dr. Brennan K. Harris, Dr. Jiaze He, Dr. Mahdi Jamshidinia, Dr. Hyunwoong Ko, Dr. Afina Lupulescu, Dr. Michael McNair, Richard Resseguie, Tristan Uribe, Dr. Mike Vasquez, and Dr. Bob Zollo.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Ethical approval No approval or endorsement of any commercial product by the National Institute of Standards and Technology is intended or implied. Certain commercial software systems are identified in this paper to facilitate understanding. Such identification does not imply that these software systems are necessarily the best available for the purpose. This material is declared a work of the U.S. Government and is not subject to copyright protection in the USA. Approved for public release, distribution is unlimited.

References

1. Blakey-Milner B, Gradl P, Snedden G, Brooks M, Pitot J, Lopez E, Leary M, Berto F, du Plessis A (2021) Metal additive manufacturing in aerospace: a review. Mater Des 209(110008):1–33

2. Kumar R, Kumar M, Chohan JS (2021) The role of additive manufacturing for biomedical applications: a critical review. J Manuf Process 64:828–850
3. Saunders S (2021) GE Aviation Announces 100,000th 3D Printed Fuel Nozzle Shipped from Auburn Plant. 3Dprint.com. https://3dprint.com/284243/ge-aviation-announces-100000th-3d-printed-fuel-nozzle-shipped-from-auburn-plant/
4. Wilkinson MD, Dumontier M, IjJ A et al (2016) The FAIR guiding principles for scientific data management and stewardship. Sci Data 3(1):1–9. https://doi.org/10.1038/sdata.2016.18
5. Bonnard R, Hascoët JY, Mognol P, Zancul E, Alvares AJ (2019) Hierarchical object-oriented model (HOOM) for additive manufacturing digital thread. J Manuf Syst. https://doi.org/10.1016/j.jmsy.2018.11.003
6. Liu C, Le Roux L, Körner C, Tabaste O, Lacan F, Bigot S (2022) Digital twin-enabled collaborative data management for metal additive manufacturing systems. J Manuf Syst. https://doi.org/10.1016/j.jmsy.2020.05.010
7. Scime L, Singh A, Paquit V (2022) A scalable digital platform for the use of digital twins in additive manufacturing. Manuf Lett. https://doi.org/10.1016/j.mfglet.2021.05.007
8. Qin Y, Qi Q, Scott PJ, Jiang X (2019) Status, comparison, and future of the representations of additive manufacturing data. Comput Aided Des. https://doi.org/10.1016/j.cad.2019.02.004
9. Mies D, Marsden W, Warde S (2016) Overview of additive manufacturing informatics: "a digital thread." Integr Mater Manuf Innov. https://doi.org/10.1186/s40192-016-0050-7
10. Chen Z, Han C, Gao M, Kandukuri SY, Zhou K (2022) A review on qualification and certification for metal additive manufacturing. Virtual Phys Prototyp. https://doi.org/10.1080/17452759.2021.2018938
11. Zhang Y, Safdar M, Xie J, Li J, Sage M, Zhao YZ (2023) A systematic review on data of additive manufacturing for machine learning applications: the data quality, type, preprocessing, and management. J Intell Manuf. https://doi.org/10.1007/s10845-022-02017-9
12. Li S, Lu Y, Aggour K, Coutts P, Harris B, Kitt A, Lupulescu A, Mohr L, Vasquez M (2023) Enabling FAIR data in additive manufacturing to accelerate industrialization. Natl Inst Stand Technol Gaithersburg MD. https://doi.org/10.6028/NIST.AMS.500-1
13. Cuddihy P, McHugh J, Williams JW, Mulwad V, Aggour KS (2018) SemTK: a semantics toolkit for user-friendly SPARQL generation and semantic data management. posters & demonstrations, industry and blue sky ideas at the 17th Interenational Semantic Web Conference (ISWC)
14. Crapo A, Moitra A (2013) Toward a unified English-like representation of semantic models, data, and graph patterns for subject matter experts. Int J Semant Comp 7(3):215–236
15. Levine L, Williams M, Zhang F, Schwalbach E, Young S, Stoudt M, Creuziger A, Borkiewicz OJ, Ilavsky J (2022) AM bench 2022 microstructure measurements for IN718 3D builds. Natl Inst Stand Technol. https://doi.org/10.18434/mds2-2692
16. Benzing J, Kafka O, Moser N, Weaver J, Liew LA, Hrabe N (2022) AM bench 2022 challenge problem subcontinuum mesoscale tensile test (CHAL-AMB2022–04-MeTT). Natl Inst Stand Technol. https://doi.org/10.18434/mds2-2587
17. Schwalbach E, Chapman M, Shah M, Uchic M, Levine L, Hrabe N, Kafka O, Moser N, Belak J (2022) AM bench 2022 IN625 3D build microstructure modeling challenge description data (AMB2022–05). Natl Inst Stand Technol. https://doi.org/10.18434/mds2-2618
18. Lu Y, Witherell P, Donmez A (2017) A collaborative data management system for additive manufacturing. In: International design engineering technical conferences and computers and information in engineering conference, vol 58110. American Society of Mechanical Engineers, p. V001T02A036
19. Lane B, Yeung H (2019) Process monitoring dataset from the additive manufacturing metrology testbed (AMMT): "three-dimensional scan strategies." J Res (NIST JRES) Natl Inst Stand Technol, Gaithersburg, MD. https://doi.org/10.6028/jres.124.033
20. Idaho National Laboratory (2021) Deep-Lynx. https://github.com/idaholab/Deep-Lynx. Accessed 16 October 2023